15.2 A Programmable 512 GOPS Stream Processor for Signal, Image, and Video Processing

Brucek Khailany[1], Ted Williams[1], Jim Lin[1], Eileen Long[1], Mark Rygh[1], DeForest Tovey[1], William J. Dally[1,2]

[1]Stream Processors, Sunnyvale, CA
[2]Stanford University, Stanford, CA

Stream processor architectures have recently emerged as fully-programmable highly-parallel processors that effectively manage data movement in computationally-demanding DSP applications [1]. This paper introduces a stream processor SoC containing a 16-lane data-parallel unit (DPU) with 5 ALUs per lane, two MIPS 4KEc CPU cores, and I/Os. Each of the 3.2mm^2 lanes run at 800MHz at 1.0V and together achieve a peak performance of 512 8b or 256 16b GOPS, or 128 billion 16b multiply-accumulates per second (128 GMACs), with a power efficiency of 82pJ/MAC (0.082 mW/MMACs). Supporting this high performance in a C-programmable processor enables flexibility and real-time processing on compute-intensive applications such as video encoding and analytics, image filtering, and wireless signal processing.

A block diagram of the stream processor SoC is shown in Fig. 15.2.1. A System MIPS manages I/O, a DSP MIPS executes application threads, and the DPU executes compute-intensive kernels. The I/Os include a memory interface, an Ethernet MAC for interfacing to an external PHY, a 33/66MHz PCI interface, an 8b parallel Flash port, configurable video ports, I2S audio ports, serial ports, GPIO, and timers. The memory interface includes two 64b DDR1/DDR2 666Mb/s memory channels for 10.7GB/s total. The video ports support several configurations: up to 9 SD or 6 HD video ports or a parallel StreamIO mode with a total of 1.5GB/s of bandwidth.

Figure 15.2.2 shows the DPU, a 16-lane implementation of a stream processing architecture [2]. Each lane contains one 16KB lane register file (LRF) and 10 VLIW function units: 5 ALUs, 1 inter-lane communication (COMM) unit, and 4 Load/Store (LdSt) units. The ALUs and COMM have 3 inputs fed by dedicated 16-word (32b) 1-read 1-write operand register files (ORFs) and two outputs that connect to all ORFs via a local switch. Each VLIW instruction encodes the ORF reads/writes, function unit operations, and control for the local switch. This instruction is broadcast to all lanes each cycle, thus supporting data-parallel operation across the lanes, and VLIW parallelism across each lane's 10 function units. The VLIW sequencer executes conditional branches and sends decoded instructions to the lanes and to a scalar unit for processing scalar variables. The instruction fetch unit loads kernel microcode into a 96KB instruction memory, storing up to 2K 384b VLIW instructions, while the stream load/store unit transfers *streams* of data between external memory and the LRF. Streams are 10s to 100s of data records defined by a base memory pointer, access pattern, and length. The DPU is controlled by stream commands sent from the DSP MIPS. Four main stream commands are supported: *Load Kernel, Execute Kernel, Load Stream, and Store Stream.* Their operands are descriptor registers for specifying memory pointers and access patterns. The DPU dispatcher buffers commands locally in a 32-entry scoreboard and dynamically issues them as resources become available, supporting concurrent memory transfers and kernel execution.

The device is programmed in C/C++; a main thread runs on the DSP MIPS with kernel function calls run on the DPU with streams of data as arguments. Kernels are compute-intensive functions such as transforms and filters that run for several microseconds processing all records from input streams in the LRF 16 records at a time. Simple C extensions and intrinsics support optimized arithmetic operations. Kernels are restricted from accessing arbitrary external memory - only addresses in the LRF associated with input and output streams are addressable, thus kernels start when all input data is available in the LRF. Streams are accessed in the LRF via one of four Ld/St units. High-bandwidth, low-latency sequential access to the LRF is implemented with pre-fetch and write buffers. An indexed Ld/St unit supports random access into the LRF [3].

Figure 15.2.3 shows how an example algorithm maps to kernels and streams. The algorithm implements the first stages of low-resolution motion search run before the more detailed integer and sub-pixel refinement. The DPU efficiency derives from the architecture's effectiveness at managing data bandwidth to minimize expensive data movement – external memory is only accessed for stream load/stores, LRF is only accessed for kernel inputs/outputs, and data is exchanged between the ORFs very often during kernels. All data movement is pre-allocated at compile-time and managed at runtime by the DPU dispatcher. There are no cache misses and runtime is deterministic, in contrast to other systems where cache-size or replacement policy can affect real-time performance.

All 80 ALUs support a variety of 2-input and 3-input operand operations found in image, video, and signal processing kernels on signed and unsigned integer datatypes with 32b, packed dual 16b, and packed quad 8b precisions. Each ALU is fully pipelined to accept one 16×32b multiply-add, two 16×16b multiply-adds or four 8×8b multiply-adds per cycle. Instruction latencies range from two cycles for logical operations to six cycles for multiplies.

Performance on a range of applications is shown in Fig. 15.2.4. High performance is achieved by extracting parallelism at four levels – concurrency between stream commands, data parallelism between the lanes, instruction-level parallelism within kernels, and sub-word SIMD parallelism in the ALU operations. The device is capable of sustaining real-time H.264 baseline profile encoding at 1080p, 30fps at high quality levels, measured within 1-2dB of the JM 10.0 baseline profile reference encoder [4]. Higher fps can be achieved by modestly relaxing quality requirements.

The full stream processor SoC shown in Fig. 15.2.7 is a 34M transistor chip implemented in a 0.13µm static CMOS standard cell library. Selectable modes support kernel operation with fewer than 16 lanes for reduced power consumption. Logic synthesis was widely used to reduce design effort, but regular structures, such as register files and multipliers, used a combination of logic synthesis, hand mapping to gates, and tiled datapath-style placement methodologies to improve routability and reduce silicon area. Most registers are full-scan, with the exception of some non-scannable flops in the embedded datapaths where full ATPG coverage could be maintained with a small serial depth. Extensive clock gating, at the SoC level, the lane level, and the per-operation level in the function units, reduced total power consumption on real applications. A hierarchical physical design style allowed careful management of local clock skew in critical paths. The 896-pin packaged SoC device has been measured to be operational from 0.8 to 1.2V using the evaluation board shown in Fig. 15.2.5. At these voltages, the DPU lanes are functional at the frequencies shown in Fig. 15.2.6.

Acknowledgements:
The authors gratefully acknowledge the contributions of U. Kapasi, S. Ermolin, K. Hesky, J. H. Ahn, J. Andrews, J. Lewis, R. Ram, B. Hsu, D. Blakely, K. Babu, B. Chen, and many other colleagues who helped work on the chip and software.

References:
[1] W. J. Dally, et al., "Stream Processors: Programmability with Efficiency," *ACM Queue,* vol. 2, no. 1, pp. 52-62, Mar., 2004.
[2] S. Rixner, et al., "A Bandwidth-Efficient Architecture for Media Processing," *Proc. of the 31st Annual Intl. Symp. on Microarchitecture,* pp. 3-13, Dec., 1998.
[3] N. Jayasena, et al., "Stream Register Files with Indexed Access," *Tenth Intl. Symp. on High Performance Computer Architecture*, pp. 60-72, Feb., 2004.
[4] AVC Fidelity Range Extensions Reference Software, ISO/IEC 14496-5:2001/Amd 8:2006, Apr., 2006.

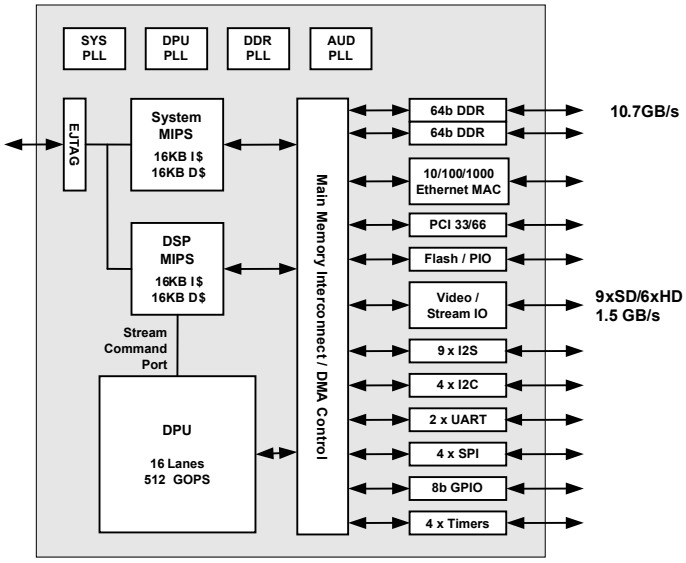

Figure 15.2.1: Stream Processor SoC Block Diagram.

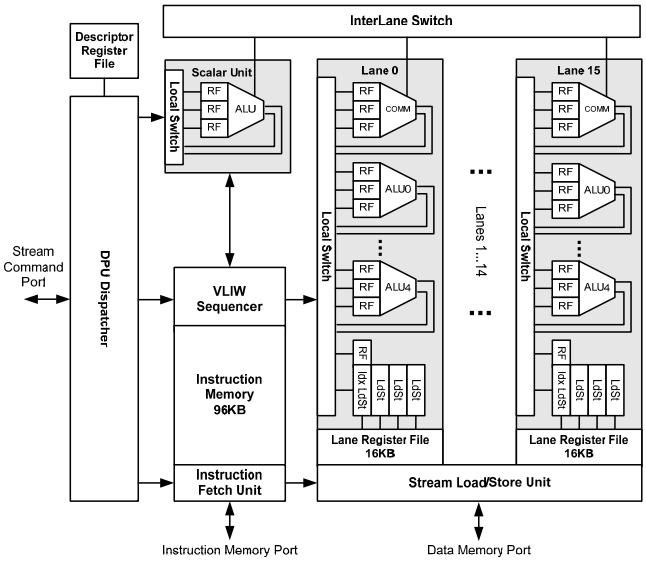

Figure 15.2.2: Data-Parallel Unit.

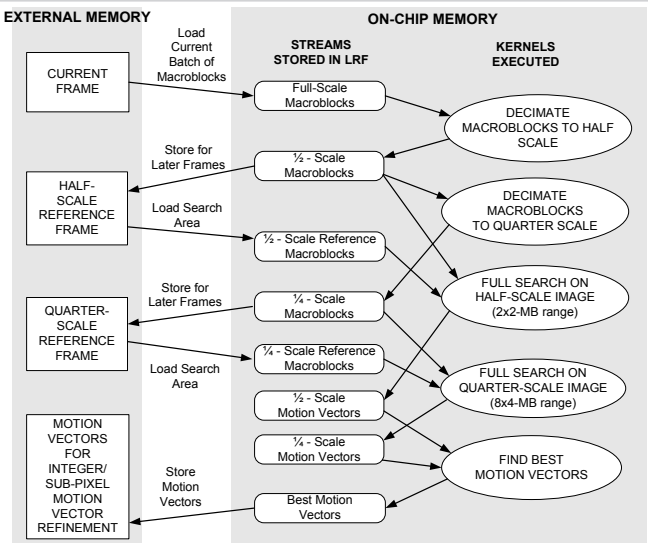

Figure 15.2.3: Kernels and Streams for H.264 Hierarchical Motion Estimation on Half-Scale and Quarter-Scale Subsampled Images.

Description	Performance	ALU Utilization
Kernel Inner-Loop Performance for General Image Processing Applications		
8x8 2-D Forward Discrete Cosine Transform	5.25 cycles / 8x8 block	82.6%
3x3 2-D Convolution Filter (8-bit pixels)	0.15 cycles / pixel	71.6%
Floyd-Steinberg Error Diffusion (8-bit monochrome)	0.89 cycles / pixel	42.5%
Kernel Inner-loop Performance for H.264 Baseline Profile Encode		
8x8 block motion estimation full search	0.72 cycles / search point	77.4%
4x4 block motion estimation full search	0.22 cycles / search point	88.6%
4x4 Forward Integer Transform and Quantization	1.8 cycles / 4x4 block	84.1%
Deblocking Filter (Luma only)	58.3 cycles / MB	64.4%
Sustained Application Performance for H.264 Baseline Profile Encode Stages		
Hierarchical Motion Estimation Low-Resolution Stages 128x64-pixel (8x4 MB) full search on quarter-scale image 32x32-pixel (2x2 MB) full search on half-scale image	372.8 cycles / MB	N/A
Deblocking Filter (Boundary Strength, Luma and Chroma)	192.9 cycles / MB	N/A

Figure 15.2.4: Kernel Inner-Loop and Application Performance Benchmarks.

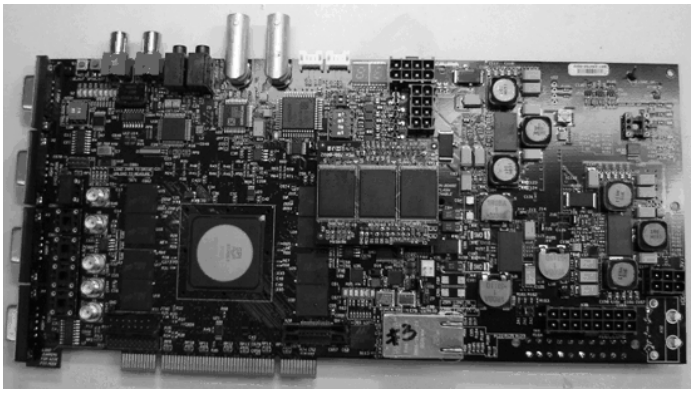

Figure 15.2.5: Evaluation Board.

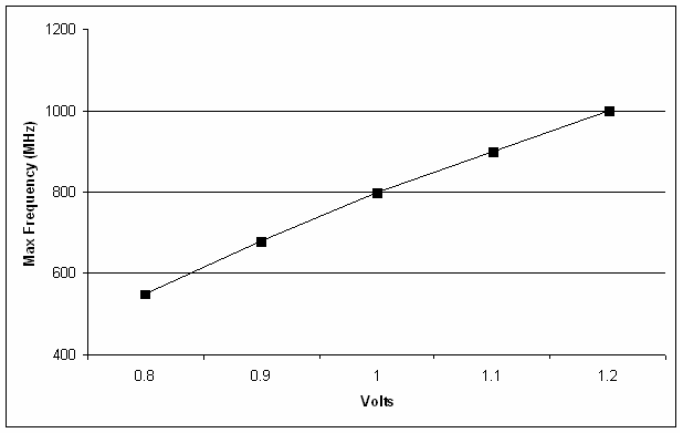

Figure 15.2.6: Measured DPU Frequency of Operation.

Continued on Page 602

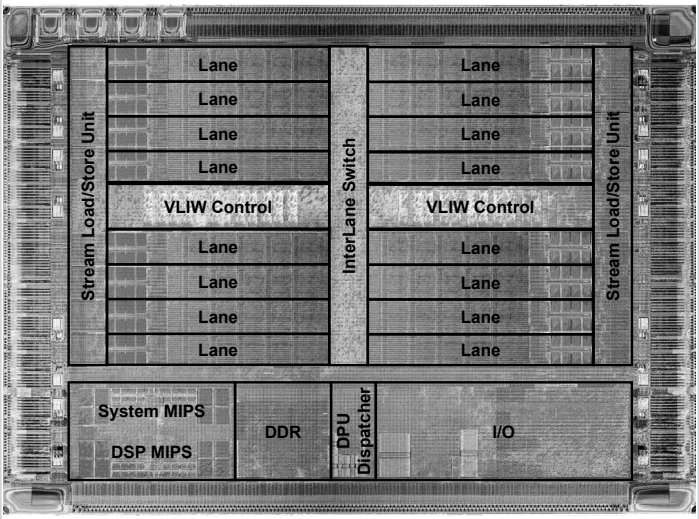

Figure 15.2.7: Die Micrograph.

